

In the claims:

**THIS PAGE BLANK (USPTO)**

1. (Original) In a content delivery system having  $m$  servers,  $S' = \{S_1, \dots, S_m\}$ ,  $n$  active customers,  $C' = \{C_1, \dots, C_n\}$ , and  $g$  geographic locations,  $G' = \{G_1, \dots, G_g\}$ , wherein  $sdel_k$  is a server delay of server  $S_k$ ,  $ndel_{j,k}$  is a network delay observed by customers in geographic location  $G_j$  while retrieving content from server  $S_k$ ,  $p_j$  is a priority value for customer  $C_i$ ,  $c_i$  is a total load of customer  $C_i$ ,  $u_{i,j}$  is a fraction of requests coming to customer  $C_i$  from region  $G_j$ ,  $a_{i,j,k}$  is a mapping representing a fraction of requests coming to customer  $C_i$  from region  $G_j$  that have been redirected to server  $S_k$ , and  $s_k$  represents a load capacity of server  $S_k$ , a method for distributing server loads, the method comprising the steps of:

representing an average prioritized observed response time as

$$AORT = \frac{\sum_{i=1}^n \sum_{j=1}^g \sum_{k=1}^m a_{i,j,k} \times u_{i,j} \times c_i \times p_i \times (sdel_k + ndel_{j,k})}{\sum_{i=1}^n c_i \times p_i}; \text{ and}$$

generating a mapping that assigns requests from customers to a particular server while minimizing  $AORT$ .

- 1                    2. (Original)   A method as recited in claim 1, further including the step of  
2   assigning all requests from all customers in all regions to a particular server such that, for  
3   each  $C_i \in C'$ ,  $G_g \in G'$ ,  $\sum_{k=1}^m a_{i,j,k} = 1.0$ .

- 1                    3. (Original)    A method as recited in claim 1, further including the step of  
2    assigning requests to a particular server while ensuring that the load capacity of each server is  
3    not exceeded such that, for each for each  $S_k \in S'$ ,  $\sum_{i=1}^n \sum_{j=1}^g a_{i,j,k} \times u_{i,j} \times c_i \leq s_k$ .

1                    4. (Original) A method as recited in claim 1, further including the step of  
2    assigning requests to a particular server while balancing the load of each server to within a  
3    maximum allowed deviation from a balanced state  $\Theta$  such that, for all pairs of servers,  $S_k$  and

$$4 \quad S_l, \frac{\sum_{i=1}^n \sum_{j=1}^g a_{i,j,k} \times u_{i,j} \times c_i}{\sum_{i=1}^n \sum_{j=1}^g a_{i,j,l} \times u_{i,j} \times c_i} \leq (1 + \Theta) \times \frac{s_k}{s_l}.$$

5. (Original) A method as recited in claim 4, wherein if the content delivery system should add one or more servers, or remove one or more customers, the load of each server will not be redistributed unless the maximum allowed deviation from a balanced state  $\Theta$  is exceeded.

6. (Original) A method as recited in claim 1, further including the step of adding one or more customers to the content delivery system only if

$$AORT_{new} \leq (1 + \Phi) \times AORT_{old} :$$

wherein  $AORT_{old}$  and  $AORT_{new}$  are old and new values of  $AORT$

$$\text{defined as } AORT_{old} = \frac{\sum_{i=1}^n \sum_{j=1}^g \sum_{k=1}^m a_{i,j,k} \times u_{i,j} \times c_i \times (sdel_k + ndel_{j,k})}{\sum_{i=1}^n c_i}$$

$$\text{and } AORT_{new} = \frac{\sum_{i=1}^n \sum_{j=1}^g \sum_{k=1}^m a'_{i,j,k} \times u_{i,j} \times c_i \times (sdel_k + ndel_{j,k})}{\sum_{i=1}^n c_i} ;$$

wherein  $a'_{i,j,k}$  is a new mapping resulting from the addition of one or more customers; and

wherein  $\Phi$  is an allowable change in  $AORT$  for existing clients.

- 1 7. (Original) A method as recited in claim 1, further including the step of
- 2 using a linear constraint solver to generate the mapping.

1                    8. (Original) A method as recited in claim 1, further including the step of  
2    using a non-linear constraint solver to generate the mapping.



1                    9. (Original) A method as recited in claim 1, further including the step of  
2    using a heuristic algorithm to generate the mapping, the heuristic algorithm comprising the  
3    step of assigning large  $a_{ij,k}$  values to small  $u_{ij} \times c_i \times (sdel_k + ndel_{j,k})$  values to produce a smaller  
4    overall *AORT* value.

1                   10. (Original) A method as recited in claim 9, the heuristic algorithm  
2 comprising the steps of:  
3                   generating a plurality of sorted lists by sorting  $C_i$  values in increasing  
4 order of  $c_i$ , sorting  $\langle C_i, G_j \rangle$  pairs in increasing order of  $u_{i,j}$ , sorting  $S_k$  values in increasing order  
5 of  $sdel_k$ , and sorting  $\langle G_j, S_k \rangle$  pairs in increasing order of  $ndel_{j,k}$ ;  
6                   starting with a top-most, smallest value item in each list, identifying  
7 comparable smallest-value items from the other lists to generate a plurality of  $\langle C_i, G_j, S_k \rangle$   
8 triples equivalent to the number of sorted lists;  
9                   selecting from the plurality of  $\langle C_i, G_j, S_k \rangle$  triples, the  $\langle C_i, G_j, S_k \rangle$  triple  
10 with the smallest  $u_{i,j} \times c_i \times (sdel_k + ndel_{j,k})$  value;  
11                   assigning to a server  $S_k$  of the selected  $\langle C_i, G_j, S_k \rangle$  triple a remaining load  
12 from the  $\langle C_i, G_j \rangle$  pair; and  
13                   repeating the heuristic algorithm starting with generating the plurality  
14 of sorted lists, taking into account the changes in the values of the  $C_i$  values and the  $\langle C_i, G_j \rangle$   
15 pairs as a result of the previous server assignment during each iteration, until the load from all  
16  $\langle C_i, G_j \rangle$  pairs has been assigned to a server  $S_k$ ;  
17                   wherein if, during any iteration of the heuristic algorithm, the load  
18 capacity of the server  $S_k$  is not sufficient to handle the remaining load, the remaining load  
19 capacity of the server  $S_k$  is assigned to some of the load of the  $\langle C_i, G_j \rangle$  pair, and an unassigned  
20 portion of the load from the  $\langle C_i, G_j \rangle$  pair is reinserted into the iterative process.

1                    11. (Original) A method as recited in claim 10, the heuristic algorithm further  
2 including the steps of:  
3                    generating a load-capacity prioritized sorted list by sorting  $C_i$  values in  
4 decreasing order of remaining load capacity  $s_k$ ;  
5                    starting with a top-most, largest value item in the load-capacity  
6 prioritized list, identifying comparable smallest-value items from the other lists to generate a  
7 load-capacity prioritized  $\langle C_i, G_j, S_k \rangle$  triple;  
8                    considering the load-capacity prioritized  $\langle C_i, G_j, S_k \rangle$  triple in the  
9 selection of the top-most  $\langle C_i, G_j, S_k \rangle$  triple with the smallest  $u_{ij} \times c_i \times (sdel_k + ndel_{j,k})$  value; and  
10                    repeating the heuristic algorithm starting with generating the plurality  
11 of sorted lists, taking into account the changes in the values of the  $C_i$  values, the  $\langle C_i, G_j \rangle$  pairs,  
12 and remaining load capacity as a result of the previous server assignment during each  
13 iteration, until the load from all  $\langle C_i, G_j \rangle$  pairs has been assigned to a server  $S_k$ .

1                   12. (Original) A method as recited in claim 10, the heuristic algorithm further  
2 including the steps of:  
3                   generating a list of content-available  $\langle C_i, S_k \rangle$  pairs in which the content  
4 of customer  $C_i$  is stored in server  $S_k$ ; and  
5                   selecting the  $\langle C_i, G_j, S_k \rangle$  triple with the smallest  $u_{ij} \times c_i \times (sdel_k + ndel_{j,k})$   
6 value that is also part of the list of content-available  $\langle C_i, S_k \rangle$  pairs;  
7                   wherein if, during any iteration of the heuristic algorithm, there is no  
8  $\langle C_i, G_j, S_k \rangle$  triple that is also part of the list of content-available  $\langle C_i, S_k \rangle$  pairs, a suitable  
9  $\langle C_i, G_j, S_k \rangle$  triple with the smallest  $u_{ij} \times c_i \times (sdel_k + ndel_{j,k})$  value is chosen and the data of  
10 customer  $C_i$  is migrated to server  $S_k$ .

1                    13. (Original) A method as recited in claim 12, the heuristic algorithm further  
2 including the step of generating a list of content-unavailable  $\langle C_i, S_k \rangle$  pairs in increasing order  
3 of migration time penalty for which the content of customer  $C_i$  is not stored in server  $S_k$ ;  
4                    wherein if, during any iteration of the heuristic algorithm, there is no  
5  $\langle C_i, G_j, S_k \rangle$  triple that is also part of the list of content-available  $\langle C_i, S_k \rangle$  pairs, a suitable  
6  $\langle C_i, G_j, S_k \rangle$  triple with the smallest combined  $u_{i,j} \times c_i \times (sdel_k + ndel_{j,k})$  value and  $\langle C_i, S_k \rangle$  migration  
7 time penalty is chosen and the data of customer  $C_i$  is migrated to server  $S_k$ .

1                   14. (Original) A method as recited in claim 1, further including the step of  
2   estimating  $ndel_{j,k}$  for non-persistent connections using HTTP logs, the step of estimating  
3    $ndel_{j,k}$  for non-persistent connections using HTTP logs comprising the steps of:  
4                   computing an estimated round trip delay  $\Delta r_{server,client}$  as  $t_{con\_req\_rec}(i+1) -$   
5    $t_{resp\_send\_end}(i)$  from information stored in the HTTP logs, where  $t_{con\_req\_rec}(i+1)$  represents a  
6   time at which a connection request message is received by the server for an  $(i+1)^{th}$  object, and  
7    $t_{resp\_send\_end}(i)$  represents a time at which the server stops sending an  $i^{th}$  object; and  
8                   computing the response time as  
9    $(t_{resp\_send\_end} - t_{con\_req\_rec}) + 2 \times \frac{\Delta r_{server,client}}{2}$ , where  $t_{con\_req\_rec}$  represents a time at which a  
10   connection request message is received by the server for an object, and  $t_{resp\_send\_end}$  represents  
11   a time at which the server stops sending the object.

1                   15. (Original) A method as recited in claim 1, further including the step of  
2   estimating  $ndel_{j,k}$  for persistent connections using HTTP logs, the step of estimating  $ndel_{j,k}$  for  
3   persistent connections using HTTP logs comprising the steps of:  
4                   computing an estimated round trip delay  $\Delta r_{server,client}$  as  $t_{con\_close\_rec} -$   
5    $t_{resp\_send\_end}(last)$ , where  $t_{con\_close\_rec}$  represents a time at which the server receives a request to  
6   close the persistent connection, and  $t_{resp\_send\_end}(last)$  represents a time at which the server  
7   stops sending a response for a last request; and  
8                   computing the response time as  
9    $(t_{con\_close\_rec} - t_{con\_req\_rec}) + \frac{\Delta r_{server,client}}{2} - \frac{\Delta r_{server,client}}{2}$ , where  $t_{con\_req\_rec}$  represents a connection  
10   request time.

1                    16. (Withdrawn)        A method for estimating a per-byte network delay  $\alpha$   
2    observed by a requesting entity in a geographic location while retrieving data from a server  
3    using TCP logs, the method comprising the steps of:  
4                    computing an *immediate* round-trip delay  $\Delta r_{imm}$  between the server and  
5    the requesting entity as  $t_{con\_est} - t_{ack\_send}$  from information stored in the TCP logs, where  $t_{con\_est}$   
6    is a time at which a connection with the server is established, and  $t_{ack\_send}$  is a time at which  
7    the server sends an acknowledgement to a connection request message received from the  
8    requesting entity; and  
9                    determining the per-byte network delay  $\alpha$  using the computed  
10   immediate round-trip delay  $\Delta r_{imm}$ , an equation  $t_{ack\_receive}(k) - t_{data\_send}(l) = \Delta r_{imm} + \alpha \times (k-l)$ , and  
11   information stored in the TCP logs, where  $t_{data\_send}(l)$  is a time at which an  $l^{th}$  byte of the data  
12   has been sent by the server, and  $t_{ack\_receive}(k)$  is a time at which acknowledgement for a  $k^{th}$  ( $k \geq l$ )  
13   byte of the data is received by the server.



1                    17 (Original) A method for estimating a response time observed by a  
2 requesting entity in a geographic location while retrieving objects from a server through a  
3 non-persistent connection using HTTP logs, the method comprising the steps of:  
4                    computing an estimated round trip delay  $\Delta r_{server,client}$  as  $t_{con\_req\_rec}(i+1) -$   
5  $t_{resp\_send\_end}(i)$  from information stored in the HTTP logs, where  $t_{con\_req\_rec}(i+1)$  represents a  
6 time at which a connection request message is received by the server from the requesting  
7 entity for an  $(i+1)^{th}$  object, and  $t_{resp\_send\_end}(i)$  represents a time at which the server stops  
8 sending an  $i^{th}$  object to the requesting entity; and  
9                    computing the response time as  
10  $(t_{resp\_send\_end} - t_{con\_req\_rec}) + 2 \times \frac{\Delta r_{server,client}}{2}$ , where  $t_{con\_req\_rec}$  represents a time at which a  
11 connection request message is received by the server from the requesting entity for an object,  
12 and  $t_{resp\_send\_end}$  represents a time at which the server stops sending the object to the requesting  
13 entity.

1                   18. (Original) A method for estimating a response time observed by a  
2 requesting entity in a geographic location while retrieving objects from a server through a  
3 persistent connection using HTTP logs, the method comprising the steps of:  
4                   computing an estimated round trip delay  $\Delta r_{server,client}$  as  $t_{con\_close\_rec} -$   
5  $t_{resp\_send\_end}(last)$ , where  $t_{con\_close\_rec}$  represents a time at which the server receives a request to  
6 close the persistent connection, and  $t_{resp\_send\_end}(last)$  represents a time at which the server  
7 stops sending a response for a last request; and  
8                   computing the response time as  
9  $(t_{con\_close\_rec} - t_{con\_req\_rec}) + \frac{\Delta r_{server,client}}{2} - \frac{\Delta r_{server,client}}{2}$ , where  $t_{con\_req\_rec}$  represents a connection  
10 request time.

19. (Original) In a content delivery network having  $m$  servers,  $S' = \{S_1, \dots, S_m\}$ ,  $n$  active customers,  $C' = \{C_1, \dots, C_n\}$ , and  $g$  geographic locations,  $G' = \{G_1, \dots, G_g\}$ , a content delivery system for distributing server loads, the content delivery system comprising:

memory for storing a server delay  $sdel_k$  of server  $S_k$ , a network delay  $ndel_{j,k}$  observed by customers in geographic location  $G_j$  while retrieving content from server  $S_k$ , a priority value  $p_j$  for customer  $C_i$ , a total load  $c_i$  of customer  $C_i$ , a fraction of requests  $u_{i,j}$  coming to customer  $C_i$  from region  $G_j$ , a mapping  $a_{i,j,k}$  representing a fraction of requests coming to customer  $C_i$  from region  $G_j$  that have been redirected to server  $S_k$ , and a load capacity  $s_k$  of server  $S_k$ ; and

a processor programmed for

representing an average prioritized observed response time as

$$AORT = \frac{\sum_{i=1}^n \sum_{j=1}^g \sum_{k=1}^m a_{i,j,k} \times u_{i,j} \times c_i \times p_i \times (sdel_k + ndel_{j,k})}{\sum_{i=1}^n c_i \times p_i}, \text{ and}$$

generating a mapping that assigns requests from customers to a particular server while minimizing  $AORT$ .

1                    20. (Original) A system as recited in claim 19, the processor further  
2    programmed for assigning all requests from all customers in all regions to a particular server  
3    such that, for each  $C_i \in C'$ ,  $G_g \in G'$ ,  $\sum_{k=1}^m a_{i,j,k} = 1.0$ .

1                    21. (Original) A system as recited in claim 19, the processor further  
2   programmed for assigning requests to a particular server while ensuring that the load capacity  
3   of each server is not exceeded such that, for each for each  $S_k \in S'$ ,

4   
$$\sum_{i=1}^n \sum_{j=1}^g a_{i,j,k} \times u_{i,j} \times c_i \leq s_k .$$

1                    22. (Original) A system as recited in claim 19, the processor further  
 2 programmed for assigning requests to a particular server while balancing the load of each  
 3 server to within a maximum allowed deviation from a balanced state  $\Theta$  such that, for all pairs

4 of servers,  $S_k$  and  $S_l$ , 
$$\frac{\sum_{i=1}^n \sum_{j=1}^g a_{i,j,k} \times u_{i,j} \times c_i}{\sum_{i=1}^n \sum_{j=1}^g a_{i,j,l} \times u_{i,j} \times c_i} \leq (1 + \Theta) \times \frac{s_k}{s_l}.$$

1                    23. (Original) A system as recited in claim 22, wherein if the content delivery  
 2 system should add one or more servers, or remove one or more customers, the processor is  
 3 further programmed for not redistributing the load of each server unless the maximum  
 4 allowed deviation from a balanced state  $\Theta$  is exceeded.

24. (Original) A system as recited in claim 19, wherein the processor is programmed for allowing one or more customers to be added to the content delivery system only if  $AORT_{new} \leq (1 + \Phi) \times AORT_{old}$ :

wherein  $AORT_{old}$  and  $AORT_{new}$  are old and new values of  $AORT$

$$\text{defined as } AORT_{old} = \frac{\sum_{i=1}^n \sum_{j=1}^g \sum_{k=1}^m a_{i,j,k} \times u_{i,j} \times c_i \times (sdel_k + ndel_{j,k})}{\sum_{i=1}^n c_i}$$

$$\text{and } AORT_{new} = \frac{\sum_{i=1}^n \sum_{j=1}^g \sum_{k=1}^m a'_{i,j,k} \times u_{i,j} \times c_i \times (sdel_k + ndel_{j,k})}{\sum_{i=1}^n c_i};$$

wherein  $a'_{i,j,k}$  is a new mapping resulting from the addition of one or more customers; and

wherein  $\Phi$  is an allowable change in  $AORT$  for existing clients.

1                    25. (Original) A system as recited in claim 19, the processor further  
2    programmed for using a linear constraint solver to generate the mapping.



1                   26. (Original) A system as recited in claim 19, the processor further  
2   programmed for using a non-linear constraint solver to generate the mapping.

1                    27. (Original) A system as recited in claim 19, the processor further  
2    programmed for generating the mapping by assigning large  $a_{i,j,k}$  values to small  $u_{i,j} \times c_i \times$   
3     $(sdel_k + ndel_{j,k})$  values to produce a smaller overall  $AORT$  value.

28. (Original) A system as recited in claim 27, the processor further  
programmed for:  
generating a plurality of sorted lists by sorting  $C_i$  values in increasing  
order of  $c_i$ , sorting  $\langle C_i, G_j \rangle$  pairs in increasing order of  $u_{i,j}$ , sorting  $S_k$  values in increasing order  
of  $sdel_k$ , and sorting  $\langle G_j, S_k \rangle$  pairs in increasing order of  $ndel_{j,k}$ ;  
starting with a top-most, smallest value item in each list, identifying  
comparable smallest-value items from the other lists to generate a plurality of  $\langle C_i, G_j, S_k \rangle$   
triples equivalent to the number of sorted lists;  
selecting from the plurality of  $\langle C_i, G_j, S_k \rangle$  triples, the  $\langle C_i, G_j, S_k \rangle$  triple  
with the smallest  $u_{i,j} \times c_i \times (sdel_k + ndel_{j,k})$  value;  
assigning to a server  $S_k$  of the selected  $\langle C_i, G_j, S_k \rangle$  triple a remaining load  
from the  $\langle C_i, G_j \rangle$  pair; and  
repeating the heuristic algorithm starting with generating the plurality  
of sorted lists, taking into account the changes in the values of the  $C_i$  values and the  $\langle C_i, G_j \rangle$   
pairs as a result of the previous server assignment during each iteration, until the load from all  
 $\langle C_i, G_j \rangle$  pairs has been assigned to a server  $S_k$ ;  
wherein if, during any iteration of the heuristic algorithm, the load  
capacity of the server  $S_k$  is not sufficient to handle the remaining load, the remaining load  
capacity of the server  $S_k$  is assigned to some of the load of the  $\langle C_i, G_j \rangle$  pair, and an unassigned  
portion of the load from the  $\langle C_i, G_j \rangle$  pair is reinserted into the iterative process.

1                   29. (Original) A system as recited in claim 28, the processor further  
2 programmed for:  
3                   generating a load-capacity prioritized sorted list by sorting  $C_i$  values in  
4 decreasing order of remaining load capacity  $s_k$ ;  
5                   starting with a top-most, largest value item in the load-capacity  
6 prioritized list, identifying comparable smallest-value items from the other lists to generate a  
7 load-capacity prioritized  $\langle C_i, G_j, S_k \rangle$  triple;  
8                   considering the load-capacity prioritized  $\langle C_i, G_j, S_k \rangle$  triple in the  
9 selection of the top-most  $\langle C_i, G_j, S_k \rangle$  triple with the smallest  $u_{i,j} \times c_i \times (sdel_k + ndel_{j,k})$  value; and  
10                  repeating the heuristic algorithm starting with generating the plurality  
11 of sorted lists, taking into account the changes in the values of the  $C_i$  values, the  $\langle C_i, G_j \rangle$  pairs,  
12 and remaining load capacity as a result of the previous server assignment during each  
13 iteration, until the load from all  $\langle C_i, G_j \rangle$  pairs has been assigned to a server  $S_k$ .

1                   30. (Original) A system as recited in claim 28, the processor further  
2 programmed for:  
3                   generating a list of content-available  $\langle C_i, S_k \rangle$  pairs in which the content  
4 of customer  $C_i$  is stored in server  $S_k$ ; and  
5                   selecting the  $\langle C_i, G_j, S_k \rangle$  triple with the smallest  $u_{i,j} \times c_i \times (sdel_k + ndel_{j,k})$   
6 value that is also part of the list of content-available  $\langle C_i, S_k \rangle$  pairs;  
7                   wherein if, during any iteration of the heuristic algorithm, there is no  
8  $\langle C_i, G_j, S_k \rangle$  triple that is also part of the list of content-available  $\langle C_i, S_k \rangle$  pairs, a suitable  
9  $\langle C_i, G_j, S_k \rangle$  triple with the smallest  $u_{i,j} \times c_i \times (sdel_k + ndel_{j,k})$  value is chosen and the data of  
10 customer  $C_i$  is migrated to server  $S_k$ .

1                   31. (Original) A system as recited in claim 30, the processor further  
2   programmed for generating a list of content-unavailable  $\langle C_i, S_k \rangle$  pairs in increasing order of  
3   migration time penalty for which the content of customer  $C_i$  is not stored in server  $S_k$ ;  
4                   wherein if, during any iteration of the heuristic algorithm, there is no  
5    $\langle C_i, G_j, S_k \rangle$  triple that is also part of the list of content-available  $\langle C_i, S_k \rangle$  pairs, the processor is  
6   further programmed for selecting a suitable  $\langle C_i, G_j, S_k \rangle$  triple with the smallest combined  
7    $u_{i,j} \times c_i \times (sdel_k + ndel_{j,k})$  value and  $\langle C_i, S_k \rangle$  migration time penalty, and migrating the data of  
8   customer  $C_i$  to server  $S_k$ .

32. (Original) A system as recited in claim 19, the processor further  
 programmed for estimating  $ndel_{j,k}$  for non-persistent connections using HTTP logs by:  
 computing an estimated round trip delay  $\Delta r_{server,client}$  as  $t_{con\_req\_rec}(i+1) -$   
 $t_{resp\_send\_end}(i)$  from information stored in the HTTP logs, where  $t_{con\_req\_rec}(i+1)$  represents a  
 time at which a connection request message is received by the server for an  $(i+1)^{th}$  object, and  
 $t_{resp\_send\_end}(i)$  represents a time at which the server stops sending an  $i^{th}$  object; and  
 computing the response time as  
 $(t_{resp\_send\_end} - t_{con\_req\_rec}) + 2 \times \frac{\Delta r_{server,client}}{2}$ , where  $t_{con\_req\_rec}$  represents a time at which a  
 connection request message is received by the server for an object, and  $t_{resp\_send\_end}$  represents  
 a time at which the server stops sending the object.

33. (Original) A system as recited in claim 19, the processor further  
 programmed for estimating  $ndel_{j,k}$  for persistent connections using HTTP logs by:  
 computing an estimated round trip delay  $\Delta r_{server,client}$  as  $t_{con\_close\_rec} -$   
 $t_{resp\_send\_end}(last)$ , where  $t_{con\_close\_rec}$  represents a time at which the server receives a request to  
 close the persistent connection, and  $t_{resp\_send\_end}(last)$  represents a time at which the server  
 stops sending a response for a last request; and  
 computing the response time as  
 $(t_{con\_close\_rec} - t_{con\_req\_rec}) + \frac{\Delta r_{server,client}}{2} - \frac{\Delta r_{server,client}}{2}$ , where  $t_{con\_req\_rec}$  represents a connection  
 request time.